Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 57 (2015) 695 – 702

Procedia Computer Science

3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

An artificial neural-network approach to software reliability growth modeling

Indhurani Lakshmanan[a]*, Subburaj Ramasamy[b]

[a] *Research Scholar, CSE Department, SRM University, Kattankulathur, Pin code: 603203, Tamilnadu, India.*
[b] *Professor, IT Department, SRM University, Kattankulathur, Pin code: 603203, Tamilnadu, India.*

Abstract

Software reliability growth models (SRGM) are statistical interpolation of software failure data by mathematical functions. The functions are used to estimate future failure rates and reliability or the number of residual defects in the software. The SRGM facilitates reliability engineers to decide when to stop testing. Although more than 200 traditional SRGMs have been proposed to estimate failure occurrence times, the research is still continuing to develop more robust models. Inherently the SRGMs are based on assumptions. In order to increase the estimation accuracy of the models we propose the SRGM based on Feed-Forward Neural Network (FFNN) approach. It seems to have significant advantages over the traditional SRGMs. Traditional parameter estimation of SRGMs need estimation ranges of parameter beforehand. The proposed artificial neural network (ANN) model does not have this requirement and hence the parameter estimation gives consistent results without any assumptions. In this paper a new neural network combination model based on the dynamically evaluated weights is proposed in order to improve the goodness of fit of already proposed traditional SRGMs and ANN based combination models. The performance comparison from practical software failure data sets seems to confirm that, the goodness of fit of proposed model is better than that of traditional SRGMs, both independent and ANN based models.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords: Software reliability growth models (SRGM); Artificial Neural Network (ANN); Software reliability estimation

* Corresponding author. Tel.: +91-805-601-0780;
E-mail address: indhurani.l@ktr.srmuniv.ac.in

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.450

1. Introduction

Software reliability is an important factor for quantitatively characterizing software quality and estimating the duration of software testing period. As per ANSI definition, software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment [12]. Software reliability models facilitate estimation of the present or future reliability of a system by estimating the parameters used in the models using software failure data at a given time. Since the year 1972, a number of stochastic software reliability growth models have been proposed. Typically there are two main categories of Software reliability models: parametric models and non-parametric models.

Parametric models estimate the model parameters based on the assumptions of underlying distributions. Parametric models can be further divided into three types : Non-Homogeneous Poisson Process (NHPP), Markovian models and Bayesian models. The NHPP models are widely used by practitioners in software reliability engineering since is also applied to hardware reliability [25]. The first continuous NHPP model was proposed by Goel and Okumoto [3] in 1979. Later, Ohba [17] presented a NHPP model with S-shaped mean value function. Yamada and Osaki [26,27] also proposed various S-Shaped NHPP models. Musa's Logarithmic Poisson Execution Time (LPET) model and Basic Execution Time (BET) [18] model, and the Kapur-Garg model [6] , imperfect debugging models proposed by Kapur and Garg [7] and Ohba and Chou [14] are some of the known NHPP models. Later on Generalized NHPP models such as Goel Generalized NHPP model [4] and Subburaj-Gopal Generalized NHPP model with GE function ROCOF [21], Generalized NHPP model with modified GE function ROCOF [22], Generalized NHPP model with shifted weibull ROCOF [23], Generalized NHPP model with modified shifted weibull function [24] were proposed. Since these NHPP or GE NHPP models depends on some assumptions , it is believed that no single model can provide accurate estimation in all situations [20]. One of the difficult tasks in parameter estimation of traditional SRGMs is estimating ranges and start values for each parameter to be estimated. Depending upon the selection of above values, the model parameters estimate may widely vary. If the analyst is not careful enough he may end up with unreasonably low or high values for each parameter but still the goodness of fit may be good enough.

On the other hand, Non-parametric models facilitates parameter estimation of the SRGMs without any presumptions. All soft computing techniques such as Artificial Neural Network, Fuzzy systems, Genetic algorithms are the non-parametric models. The problems with the parameter estimation of traditional SRGM is overcome by artificial neural network (ANN) combination model which is non-parametric. When we use ANN there is no need to specify the range of values in advance for each parameters which is a complex task. The influence of external parameters and their assumptions of a model can be eliminated when we design a model that is able to evolve itself based on the software failure data. ANN improves the parameter estimation paradigm. Also non-parametric methods can produce models with better predictive accuracy than parametric models [9,10,20]. Karunanithi et al [9,10] first used neural networks to predict software reliability by using the execution time as input and the cumulative number of detected faults as the desired output. Sitte [19] compared neural networks and recalibration for parameter models to predict the reliability by using common predictability measure and common datasets. Cai et al [1] proposed the effectiveness of the neural back-propagation network method (BPNN) for software reliability prediction by using the multiple recent inter-failure times as input to predict the next failure time. Su and Huang [20] presented a dynamic weighted combinational model for software reliability prediction based on neural network approach. Jun Zheng [5] examined a single neural network with three parametric NHPP models for software reliability prediction. Wang and Li [2] combined the classical software reliability models and neural network to improve the accuracy of software reliability prediction. Roy et al [16] combined feed forward and recurrent neural network for software reliability prediction.

In this paper, we propose a FFNN by combining two of the available Generalized NHPP software reliability growth models using ANN approach. The traditional models are used as the base models and the neural networks are used to combine the base model. The classical SRGMs are merged based on the dynamically evaluated weights determined by the Back propagation training method of the proposed FFNN. We compare the performances of the proposed model with the base models and also with the already proposed neural network combination model [2] with two practical software failure data sets.

This paper is organized as follows: Section 2 describes the neural network components which are used to construct the proposed artificial neural network architecture. Section 3 presents the proposed FFNN approach based on the dynamic weighted combination model for software reliability estimation. Section 4 gives the performance evaluation based on two practical software failure data sets of the two base traditional models of the proposed model, already proposed ANN based combination model and proposed combination model. Summary and conclusion are given in section 5.

2. Neural Network Components

A neural network is a network of interconnected nonlinear neurons inspired from the studies of the biological neuron system. The function of a neural network is to produce an output pattern when presented with an input pattern. An artificial neural network (ANN) has a parallel-distributed architecture with a large number of nodes (neurons) and connections. The proposed neural network combination model is constructed by three basic components as follow:

- Neurons : A neuron is an information processing unit that is the basis to the operation of a neural network. each neuron can receive input signal, process the signals by using activation function and finally produce an output signal.
- Network Architecture : The simplest and most common type of neural network architecture is called Feed-Forward Neural Network (FFNN). The neurons are organized in the form of three layers: an input layer, a hidden layer, and an output layer. Circles represents the neurons and the connection of neurons across layers is called the connecting weight. The data and calculation flows in a single direction, from the input data to the outputs through hidden neurons.
- Learning algorithm : The neural network is trained through learning algorithm by providing network with a series of sample input and comparing the expected sample output with the desired response over a specified period of time by adjusting weights. The back-propagation algorithm is the most widely used learning method for multi-layer feed-forward networks. The name refers to the backward propagation of error during the training of the network which is produced by comparing the output of the network with a desired response. This error signal is propagated through the network in the backward direction by adjusting weights of the network to optimize the network performance. The training procedure carried out until network is able to provide expected responses.

By using the above three neural network components, the output of the neural network in a mathematical term is defined as

$$y = f(s) \quad \text{and} \quad s = \sum_{j=1}^{N} wjxj \tag{1}$$

where
N is number of input elements i.e., x1, x2, x3, ……. xN
f() = activation function processes the input signals and produces the final output of the neuron. It also limits the range of output of the neuron so that the output never exceeds the limits.
Σ = summation function which sums the input signals with the respective weights of the input elements.
wj = w1, w2,w3 , …….wN are the respective weights of input elements and
y = output of the neural network.

3. Proposed ANN for Software reliability growth modeling

3.1. The selection of base models

We construct the neural network-based combination model with single input neuron in the input layer, single output neuron in the output layer and two neurons in the hidden layer. The number of neurons in the hidden layer is

determined by the number of base models selected to construct the neural network combination model. In the past, the ANN based combination models were proposed by the researchers with two parameters NHPP as base models such as GO model, Such model, Delayed S-Shaped model and logistic growth model [1,2,5,20]. The three parameter generalized NHPP models were chosen in the proposed model because of their ability to provide consistent goodness of fit, in spite of wide fluctuations in data and whether the mean value function depicts exponential growth or S-shaped growth [21].

Figure 1. represents the proposed feed forward neural network combination model based on dynamically evaluated weights using back propagation training. The mean value functions of the base models are as follows.

Mean value function of Goel Generalized NHPP(G-O GE NHPP) model

$$f(x) = a\left(1 - e^{-bx^c}\right) \tag{2}$$

Mean value function of Subburaj-Gopal Generalized NHPP (S-G GE NHPP) model

$$f(x) = a\left(1 - e^{-x/b}\right)^c \tag{3}$$

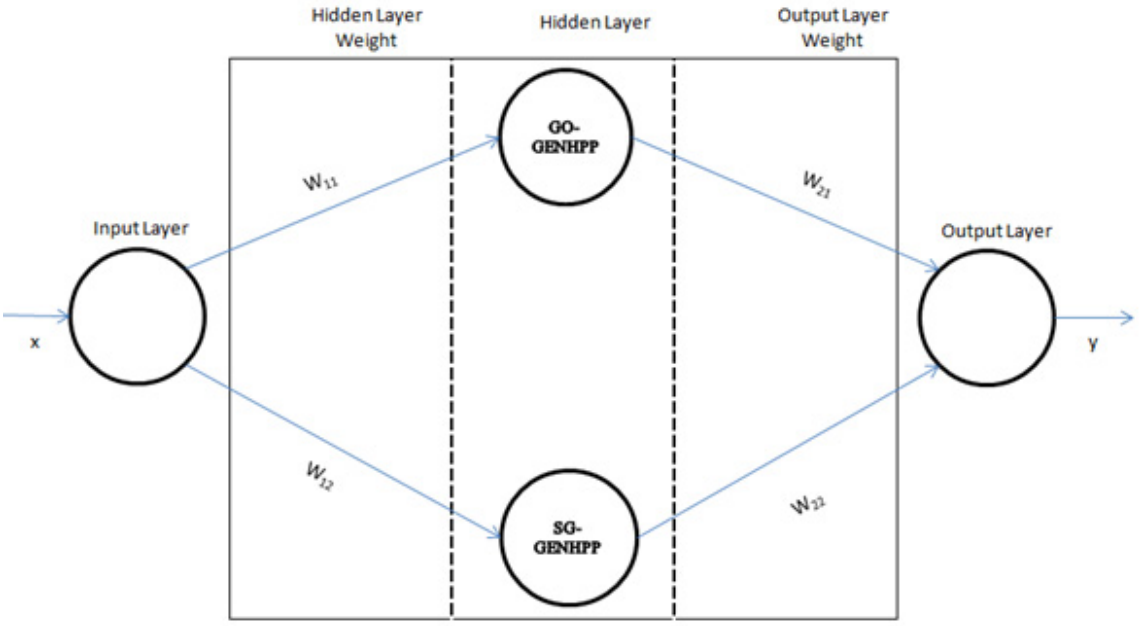

Figure 1. Proposed Feed Forward Neural Network Combination Model

The output of the proposed neural network combination model is as follows

$$Y(x) = w21\left(1 - e^{-w11x^{c1}}\right) + w22\left(1 - e^{-x/w12}\right)^{c2} \tag{4}$$

where w1j,w2j(>0) are the weights of the FFNN and their values are determined by the training algorithm. Here, c1,c2(> 0) are activation function parameters whose values are also evaluated through the learning of the proposed FFNN.

3.2. The implementation and derivation of the proposed ANN combination model

This section describes the implementation and derivation of the proposed neural network based combination model. The implementation steps that are required to construct the neural network based combination model are as follows.

- analyze the characteristics of software project and failure data set, and select the appropriate software reliability model as the base models
- construct the neural network by designing a proper activation function for each base model
- feed the software failure data sets to the network to train the network by the back-propagation algorithm
- estimate the parameter and predict the software reliability with the designed neural-network-based combination model.

According to the requirements for the activation functions, the activation functions should be designed under some conditions for the mean value functions of selected base models. For back propagation feed forward neural network model, the activation function should be 1.simple 2. differentiable 3. continuous everywhere and the output function can be approximately expressed as the form of composite function g(f (x)). It can be proved that the activation function f(x) and g(x) is continuous and differentiable everywhere, so f(x) and g(x) are in accordance with the conditions of the activation function.

We can derive a form of compound functions from its mean value function as the following:

Mean value function of G-O Generalized NHPP model.

$$f(x) = a\left(1 - e^{-bx^c}\right)$$

Assume that

$$f(x) = 1 - e^{-bx^c} \text{ and } g(x) = ax$$

Therefore, we can get

$$g\left(f(x)\right) = g\left(1 - e^{-bx^c}\right) = a(1 - e^{-bx^c})$$

This means that mean value function of G-O Generalized NHPP model is composed of

g(x), f(x).

After the derivation above, we find that if we assume the activation functions f(x) and g(x) as:

$$f(x) = 1 - e^{-x} \tag{5}$$

$$g(x) = x \tag{6}$$

If we construct a neural network with an activation function as $1 - e^{-x}$ in the hidden layer, with the weights w11 and w21. Then we have the outputs of the hidden layer and output layer are as follows.

$$h(t) = 1 - e^{-w21x^c} \tag{7}$$

and

$$y(t) = w11(1 - e^{-w21x^c}) \tag{8}$$

Compare (2) and (8), It is G-O Generalized NHPP model and w11 = a and w21 = b.

Following the similar procedure, we can easily extend this concept to construct a neural network with appropriate activation functions and corresponding weights for a given SRGM.

4. Performance Evaluation

4.1. Software failure data

The software failure data are arranged in pairs {x,y} where x is the cumulative test time and y is the corresponding cumulative number of failures. In this experiment, two practical software failure data sets [13] DS1and DS2 are used to check the performance of the proposed feed forward neural network based combination model.

The description of the datasets are given below.

- Data Set-1 (DS1): This data set was collected by John D. Musa of Bell Telephone Laboratories based on the 136 failures observed from a real-time command and control system with 21,700 lines of instructions.
- Data Set-2 (DS2): This data set was collected by Musa for a military application which consists of 1,80,000 lines of instructions with 101 failures.

In many testing situations, it has been observed that the failure rate first increases and then starts decreasing (I/D) due to the learning phenomenon of the testing team [21]. When faults are corrected without introducing new faults, the successive failures occur at times t1, t2,...,tn, such that 0<t1<t2<...<tn . The successive inter-failure times namely, (t2 - t1), (t3 - t2) etc. should be such that: (t2 - t1) < (t3 - t2) < (t4 - t3) < (t5 - t4)…

When the above relationship is violated occasionally, we may say that there are mild fluctuations in time between failures data. When it occurs too frequently, we say that there are wide fluctuations. This occurs due to dynamics of the software projects and cannot be avoided in practice and we need models that will perform even when there are reasonable fluctuations in data.

We took one data set-1 (DS1) with mild fluctuations and another one data set-2 (DS2) with wild fluctuations due to the above reason. The first 30% of the data were used for training the model, and the remaining data were used to test the model.

4.2. Model performance measures

In order to compare the performance of software reliability models, the researchers proposed a number of criteria to evaluate the models. We measure the performance of the proposed model by using R^2 (Coefficient of determination) , RMSE (Root Mean Square Error) and AIC (Akaike Information Criterion). The value of R^2 may vary from 0 to 1. The closer R^2 is to 1, the better is the fit. RMSE is the criterion to measure the difference between the actual and predicted values and is a more general metrics [11,12], . AIC is a measure of goodness of fit of an estimated statistical model. AIC is considered to be a measure which can be used to rank the models and it gives a penalty to a model with more number of parameters. RMSE and AIC are computed as follows:

$$\mathrm{RMSE} = \sqrt{1}/\mathrm{k}\sum(\mathrm{P}-\mathrm{A})^2 \tag{9}$$

where k is the number of observations, P is estimated error and A is actual error observed during testing phase. The smaller RMSE indicates less fitting error and better performance.

$$\mathrm{AIC} = \mathrm{n}\log\left(\frac{\mathrm{RSS}}{\mathrm{n}}\right) + 2\mathrm{N} \tag{10}$$

where n is the number of observations, RSS is residual sum squares and N is the number of parameters in the model. Minimum AIC value represents the best fit model.

4.3. Performance analysis

we analyze the performances of our proposed feed forward neural network combination model with two selected generalized exponential SRGMs. We use the least square estimation to solve the parameters. The proposed dynamic weighted neural network combination model which can combine the SRGMs based on the dynamically assigned weights determined by the training of the proposed ANNs. Then we compare our proposed neural network combination model (NN-2) with the already proposed neural network combination model (NN-1) and two selected base traditional models (M1, M2)of the proposed FFNN model.

Table 1. COMPARISONS OF R2 RMSE AIC

	M1			M2			NN1			NN2		
	R^2	RMSE	AIC	R^2	RMSE	AIC	R^2	RMSE	AIC	R^2	RMSE	AIC
DS-1	0.9901	3.926	18.31	0.9911	2.539	16.59	0.9938	0.2953	5.92	0.9958	0.0952	4.31
DS-2	0.9891	5.628	25.61	0.9901	3.592	22.81	0.9917	0.9692	10.38	0.9967	0.1478	5.37

M1 : G-O Generalized NHPP model
M2 : S-G Generalized NHPP model
NN1 : Neural Network Combination model with the base models G-O model, Delay S-shape model and Sch model
NN2 : Proposed Neural Network Combination model with the base models G-O Generalized NHPP and S-G Generalized NHPP

The summary of R^2, RMSE and AIC can be depicted from Table 1. It can be seen that the Neural Network model (NN-1,NN-2) give better fit than classical models (M1,M2). Next, by comparing NN-1 and NN-2, the NN2 model has better than NN-1. Note that NN-2 model has larger improvement than the NN-1 model for DS-2. As we mentioned in the previous section, the data set-2 (DS-2) has wider fluctuations due to the learning phenomenon of testing team as compared to DS-1 as revealed by all the goodness of fit indices. NN-1 base models are sensitive to wild fluctuations in data. But the base models in NN-2 that will perform even when the learning phenomenon occurs and there are reasonable fluctuations in data. AIC is considered to provide overall performance index of a model for a given dataset. Table 1 provides the ranking for the performance of the models for the chosen dataset.

5. Conclusion

In this paper we propose a new feed forward neural network based dynamic weighted combination model using back-propagation algorithm as an SRGM to improve the software reliability estimation accuracy. The performance comparison using practical software failure data sets show that, the proposed model estimation accuracy is better than that of traditional SRGMs and already proposed neural network combination model. All the four SRGMs were evaluated using two practical software failure data sets with varying characteristics. This paper also will confirm that if the base models are more flexible then we get a better SRGM based on ANN.

Acknowledgements

This work was supported by the Department of Science and Technology, India under Grant DST/2013/849.

References

[1] Cai KY, Cai L, Wang WD, Yu ZY, and Zhang D. On the neural network approach in software reliability modeling, *J Systems and Software* 2001;58; 47–62.
[2] Gaozu Wang and Weihual Li. Research of Software reliability combination model based on neural network, *IEEE second WRI world congress on Software Eng* 2010.
[3] Goel AL and Okumoto K. Time-Dependent error-Detection Rate Model for Software Reliability and Other Performance Measures, *IEEE trans on Reliab* 1979; 28(3);206-211.

[4] Goel AL. Software reliability models: Assumptions, limitations and applicability, *IEEE Trans on Software Eng* 1985; SE-11 (12);1411-1423.
[5] Jun Zheng. Predicting software reliability with neural network ensembles, *Expert Systems with Applications* 2009;36; 2116-2122.
[6] Kapur PK and Garg RB. Optimal software release policies for software reliability growth models under imperfect debugging, *Operations Research* 1990;24;295-305.
[7] Kapur PK and Garg RB. A software reliability growth model for an error removal phenomenon, *Software Eng J* 1992;7(4);291-294.
[8] Kapur PK, Archana Kumar, Rubina Mittal and Anu Gupta. *Flexible Software Reliability Growth Model Defining Errors of Different Severity, Reliability, Safety and Hazard*, Editors: P.V.Varde et al., Narosa Publishing House, New Delhi, India, 2006.
[9] Karunanithi N, Whitley D and Malaiya YK. Using neural networks in reliability prediction, *IEEE Software* 1992;9;53–59.
[10] Karunanithi N, Whitley D and Malaiya YK. Prediction of software reliability using connectionist models, *IEEE Trans on Software Eng* 1992;18;563–574.
[11] Karunanithi N and Malaiya YK, The Scaling Problem in Neural Network for Software Reliability Prediction, *Proc of the Third International Symposium on Software Reliab Eng,* IEEE Computer Society Press 1992;76-82.
[12] Lyu Michael R. *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, McGraw Hill, New York,1996.
[13] Musa JD. DACS Software Reliability Dataset, *Data & Analysis Center for Software*, January 1980.
[14] Musa JD and Ackerman AF. Quantifying software validation: When to stop testing?, *IEEE Software* 1989;19-27.
[15] Musa JD, Iannino A and Okumoto K. Software Reliability, *Measurement, Prediction and Application*, McGraw-Hill, 1987.
[16] Pratik Roy, Mahapatra GS, Pooja Rani, Pandey SK and Dey KN, Robust feed forward and recurrent neural network based dynamic weighted combination models for software reliability prediction, *Applied Soft Computing* 2014;22;629-637.
[17] Ohba M, et al., S-shaped Software Reliability Growth Curve: How Good Is It?, *COMPSAC'82* 1982;38-44.
[18] Ohba M and Chou XM. Does imperfect debugging affect software reliability growth?, *Proc. 11th International Conf. Software Eng,* 1989;237-244.
[19] Sitte R. Comparison of software-reliability-growth predictions: neural networks vs parametric recalibration, *IEEE Trans on Reliab* 1999;48(3);285 - 291.
[20] Su YS and Huang CY. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models, *J of Systems and Software* 2007;80;80.
[21] Subburaj R and Gopal G. Generalized Exponential Poisson Model for software reliability growth, *International J of Performability Eng* 2006;2(3);291-301.
[22] Subburaj R, Gopal G and Kapur PK. A Software Reliability Growth Model for Vital Quality Metrics, *South African J of Industrial Eng* 2007;18 (2);18.
[23] Subburaj R and Gopal G. Software Reliability Growth Model Addressing Learning, *J of Applied Statistics* 2008;35(10);1151 – 1168.
[24] Subburaj R, Gopal G and Kapur PK. A Software Reliability Growth Model for Estimating Debugging and the Learning Indices, *International J of Performability Eng* 2012;8(5); 539- 549.
[25] Xie M. Software Reliability Modeling, *World Scientific*, Singapore, 1991.